Algorithmen und Programmieren 3 **Matrikelnummer:** ____________
WS 2003/2004, Zwischenklausur **NAME:** ____________________

Abgabe bis Dienstag, 16. Dezember 2003

Anleitung: Streichen Sie *eine* Aufgabe deutlich auf dem Angabeblatt. Diese Aufgabe wird nicht in die Bewertung einbezogen. Wenn Sie selbst keine Aufgabe streichen, wird die erste Aufgabe nicht in die Bewertung einbezogen. Bearbeiten Sie die übrigen drei Aufgaben.

Schreiben Sie jede Aufgabe auf ein getrenntes Blatt. Jede Aufgabe hat 10 Punkte.

---

1. Ein binärer Suchbaum soll Einträge speichern, die außer dem Schlüssel x_i noch ein „Gewicht“ w_i enthalten. Die Algorithmen zum Suchen, Einfügen und Löschen in binären Suchbäumen sollen so erweitert werden, dass man jederzeit zu einem Wert a das Gesamtgewicht aller Einträge finden kann, deren Schlüssel $x < a$ ist.

   Die Laufzeit für alle Operationen soll höchstens proportional zur Höhe des Baumes sein.

   Beschreiben Sie die Felder, die in den Knoten enthalten sind (zusammen mit einer Erklärung der Bedeutung, sofern sie nicht offensichtlich ist), und ein Programmstück (eine oder mehrere Methoden in Java oder Funktionen in Haskell), das die Intervallabfrage durchführt. (Einfügen und Löschen müssen Sie *nicht* programmieren.)

2. Programmieren Sie eine *stabile* Variante von Sortieren durch Verschmelzen (*mergesort*) in Java oder Haskell. Erläutern Sie die Stellen im Programm, die für die Stabilität wichtig sind.

3. Sortieren durch Zählen:
   Das folgende Programmstück soll n verschiedene `double`-Zahlen sortieren, die in einem Array `a[0]`, …, `a[n-1]` gespeichert sind, mit Hilfe eines Arrays `pos[i]`, das die Anzahl der Elemente zählt, die kleiner als `a[i]` sind.

```
int [ ] pos = new int[n];
for (int i=0; i<n; i++) { // Zähle Elemente kleiner als a[i]:
    pos[i] = 0;
    for (int j=0; j<n; j++)
        if (a[j]<a[i]) pos[i]++;
    // pos[i] ist die Position von a[i] in der sortierten Reihenfolge.
}
// In die richtige Reihenfolge umordnen:
for (int i=0; i<n; i++) a[pos[i]] = a[i];
```

   (a) Was ist der Speicherbedarf und die asymptotische Laufzeit dieses Programmes?

   (b) Warum funktioniert dieses Programm nicht richtig? Geben Sie ein Beispiel an, bei dem das Array `a[0]`, …, `a[n-1]` am Ende *nicht aufsteigend sortiert* ist.

   (c) Stellen Sie das Programm richtig, sodass am Ende die sortierten Zahlen in `a[0]`, …, `a[n-1]` stehen, ohne dass die die asymptotische Laufzeitschranke in Aufgabe (a) überschritten wird.

   (d) Was passiert, wenn die Eingabe gleiche Werte enthält?
   Erstellen Sie eine Variante des Programms, die auch solche Eingaben sortiert.

4. Beweisen Sie: Ein binärer Baum mit n Blättern hat mindestens $n-1$ innere Knoten. (Innere Knoten haben 1 oder 2 Kinder, und Blätter haben keine Kinder.) Sie dürfen nicht einfach die Aussage von Übung 31 über volle binäre Bäume verwenden, aber Sie dürfen die Lösung (den Beweis) dieser Aufgabe anpassen.